**Graduate Course on Embedded Control Systems: Theory and Practice**

**Networks for Embedded Control Systems**

Luis Almeida – University of Porto
Paulo Pedreiras – University of Aveiro

RETIS Lab
Scuola Superiore Sant’Anna, Pisa
8-12 June 2009

---

**Outline**

- **Trends in embedded systems**
  - Distribution, integration and operational flexibility
- **Timing issues in the network**
  - Timing parameters, time across the network, temporal control of the communication
- **Inside the protocol stack**
  - The physical, data link, (network) and application layers
- **A few related protocols**
  - WorldFIP, TTP/C, PROFIBUS, CAN, Ethernet, FlexRay, WiFi, ZigBee
- **Some on-going research topics**

---

**Speakers background**

- **Real-time systems:**
  - Scheduling aspects (processor, network, holistic,...)
  - RT kernels (mainly for small micro-controllers)
- **Dependable systems:**
  - Architecture of embedded systems
  - Safety and reliability aspects
  - Safety-critical systems
- **Main battle fields:**
  - Dynamic reconfiguration
  - On-line QoS adaptation
  - Flexible scheduling

---

**Towards distribution**

Nowadays, complex embedded systems are **distributed**, with a network connecting several active components

- Cars, trains, planes, industrial machinery ...

---

**Dissemination of feedback control**

The number of **automatic control loops** within complex embedded systems increased dramatically motivated by

- Improved **performance** (e.g. TCS – Traction Control)
- Increased **safety** (e.g. ABS, ESP – Elect. Stability Prog.)
- Increased **energy efficiency** (e.g. Engine control)
- New **functionality** (e.g. Autonomous parking)
- More **comfort** (e.g. Cruise control)

Most of such loops are distributed leading to **Distributed Computer Control Systems (DCCS)**
Towards functional integration

Integrating several quasi-independent subsystems allows further performance improvements:
- E.g. integrating ABS, ESP, TCS and Active Suspension might significantly improve car overall stability control

Using smart components (sensors and actuators) greatly simplifies higher control levels and facilitates functional integration.

This is sometimes referred to as centralized control, but still generally using a distributed HW architecture!

Integration and control

Centralized control is a form of hierarchical control:
- Integrating the three wheels control in holonomic robots provides full holonomic motion (any direction with any orientation)
- Visual tracking uses holonomic system as a smart actuator

Towards operational flexibility

Many DCCS operate in dynamic scenarios:
- Systems with variable number of users or variable load (traffic control, radar...)
- Systems that operate in changing physical environments (robots, cars...)
- Systems that can self-reconfigure dynamically to cope with hazardous events or evolving functionality (cars, planes, trains, production cells...)

QoS adaptation, graceful degradation, survivability

Towards sensing diffusion

- Controlling large processes
- Involving sensing in large areas
  - Precision agriculture, environment control, ...
  - Cyber-Physical Systems
  - Wireless Sensor Networks

Optimizing water use in golf courses
Putting it all together

Merging distribution, control, integration and flexibility raises many issues that need adequate treatment

- Higher potential for mutual interference
  - Higher jitter in periodic executions
  - Longer delays (sampling→control→actuation)
  - Information losses (errors, vacant sampling issues)
  - Control performance degradation
  - Control instability...
- Conflict between safety and flexibility
  - Flexibility reduces a priori knowledge
  - Requires adequate temporal isolation and temporal control

Network delays

\[ t_{d1} = t_{q1} = t_{l} \text{, network-induced delay} \]
\[ d = q_{l} + d_{l} = d_{1} = t_{q} \text{, delay components} \]
\[ d = d_{1} = j \text{, delays jitter} \]

Delay and delay jitter

- **Network induced delay** — extra delay caused by the transmission of data over the network.
  - Some applications (e.g. control) are particularly sensitive to this.
- **Delay jitter** — variation affecting the network delay.
  - Some applications (e.g. streaming) are not much affected by the network delay but are highly affected by delay jitter

Burstiness and buffering

- **Burstiness** — measure of the load submitted to the network in a short interval of time.
  - Bursts have a profound impact on the real-time performance of the network and impose high buffering requirements.

- **Buffer requirements** — Buffer capacity needed to hold packet bursts.
  - Too few buffers may lead to packet losses

Throughput and arrival / dep. rate

- **Throughput** — average amount of data, or packets, that the network dispatches per unit of time (bit/s and packets/s).
- **Arrival / departure rate** — long-term rate at which data arrives at/from the network (bit/s and packets/s).

\[ \text{Throughput} \leq \text{Network capacity} \]

\[ \text{(e,p)-model (Chiu, 1991)} \]
Throughput versus timeliness

- **Throughput** equates to speed, e.g., average number of packets delivered per unit of time
- **Timeliness** is related to the instant of executing certain computations (not directly related to speed)
- **Timeliness** is highly influenced by the scheduling of activities in the system, i.e., when several packets are ready for transmission, which is sent first?

Time across a network

- However, a coherent notion of time can be very important for several applications to:
  - Carry out actions at desired time instants
    - e.g., synchronous data acquisition, synchronous actuation
  - Time-stamp data and events
    - e.g., establish causal relationships that led to a system failure
  - Compute the age of data
  - Coordinate transmissions
    - e.g., TDM-based systems
  - etc.
- But how to synchronize the clocks across the network?

Clock synchronization

Few definitions ($C_p(t)$ is the clock of node $i$ at instant $t$)

- Accuracy $\alpha$
  - $|C_p(t) - t| \leq \alpha$ for all $i$ and $t$
- Precision $\delta$
  - $|C_p(t) - C_p(t)| \leq \delta$ for all $i, j, t$
- Offset
  - Difference between $C_p(t)$ and $t$
- Drift
  - Difference in growing rate between $C_p(t)$ and $t$

Synchronizing clocks

- Clocks can be synchronized:
  - **Externally** – an external clock source sends a time update regularly (e.g., GPS)
  - **Internally** – nodes exchange messages to come up with a global clock value
  - **Master-Slave** – A special node (time master) spreads its own clock to all other nodes
  - **Distributed** – all nodes perform a similar role and synchronize their clocks using the clocks of the others, for example, using an average (e.g., FTA, Fault-Tolerant Average)
- Standards: NTP, SNTP, IEEE 1588

Real-time messages

- A message related to a real-time entity (e.g., a sensor) is a **real-time message**.
- Real-time messages must be transmitted within precise time-bounds to assure coherence between senders and receivers concerning their **local views** of the respective real-time entities
- Real-time messages can have event or state semantics
Event or Time triggering
✓ According to the type of messages (event or state) conveyed by the network, it can be:
  ✓ event-triggered (event messages)
  ✓ time-triggered (state messages)

Event triggered network
✓ Transactions, carrying event data, are triggered upon event occurrence.
✓ Consequently, transmission instants are generally asynchronous with the nodes or network.
✓ Receivers know about system state by means of the received events.
✓ The submitted communication load (number of simultaneous message transmission requests) may be very high (event showers).

Time triggered network
✓ There is a notion of network time (explicit or implicit)
✓ Transactions, carrying state data, are triggered at predefined time instants.
✓ Receivers have a periodic refresh of the system state
✓ The submitted communication load is well determined.

Event vs time triggering
What is more important?!
- Uses global a priori knowledge
  - Facilitates fault-tolerance
  - Prompt omission detection
  - Simple backward recovery
- Complex deployment
- Long omission detection
- Complex fault-tolerance

What about both?
Transmission control

- Determines who triggers network transactions, application or network
  - External control: Transactions are triggered upon explicit control signal from the application. Messages are queued at the interface. Highly sensitive to application design faults.
  - Autonomous control: The network triggers transactions autonomously. No control signal crosses the CNI. Applications exchange data with the network by means of buffers. Deterministic behavior.

Information flow

- Determining end-to-end delay
  - ET-network with external control: Transactions are composed of several elementary actions carried out in sequence.
  - TT-network with autonomous control: The elementary actions in each intervenient (transmitter, network, receiver) are decoupled, spinning at an appropriate rate.

Information flow

- In a TT-network
  - The tight control of the relative phase required between all system activities (message transmissions and task executions) imposes rigid architectural constraints.
  - Time-triggered architecture: The whole system must be designed altogether (network and nodes).
  - However, once the network is designed, nodes will not interfere with each other (transmissions occur in disjoint intervals).
  - Composability with respect to temporal behavior:

Interoperability

- Exchanging information and cooperating with other systems (nodes)
  - Nodes in a distributed system must be interoperable
  - How to assure that different nodes, with CNIs and network controllers of different makers will be interoperable?

Part 3
Inside the protocol stack

OSI reference model

- The Open Systems Interconnection (OSI) reference model
  - Developed within ISO from 1977 to 1984
  - Became International Standard IS 7498
  - It is the reference model used in the development of interconnection and cooperation standards for open distributed systems
Node A

OSI 7 layers
reference model

Network
Data Link
Transport
Session
Presentation
Application
Nodes A/B

Large computing and communication overhead for short data items

Protocol stack

Node B

OSI collapsed model

- Application services access the Data Link directly
- Other layers maybe present but not fully stacked
- In process control and factory automation these networks are called Fieldbuses

Multi-hop networks

- Due to physical and/or logical constraints many control applications cannot be supported by local networks
- (Wireless) sensor networks, complex industrial plants...
- A network layer with routing services is fundamental (multi-hop communication)
- Some more complex application layers, Specific middlewares with certain services
  - e.g., localization, tracking, data aggregation
- Energy-aware / location-aware routing and traffic scheduling
- Synchronization becomes particularly important
  - e.g., for channel reuse

Physical layer

- Issues related with the physical layer:
  - Interconnection topology
  - Physical medium
  - Coding of digital information
  - Transmission rate
  - Maximum interconnection length
  - Max number of nodes
  - Feeding power through the network
  - Energy Consumption
  - Immunity to EMI
  - Intrinsic safety

Physical layer

- Interconnection topology
  - Tree
  - Mesh (wired)
  - Star
  - Bus
  - Mesh (wireless-RF)
  - Ring
### Physical layer

**Topologies**

<table>
<thead>
<tr>
<th>Topology</th>
<th>In favor</th>
<th>Against</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mesh</td>
<td>Point-to-point connections (predictable, scalable), several alternative paths</td>
<td>Require routing, complex cabling (need to avoid logical order)</td>
</tr>
<tr>
<td>Tree (star)</td>
<td>Point-to-point connections, broadcast communication in parallel branches</td>
<td>Require routing, potential long paths for deep nodes or different branches, upper branches are bottlenecked</td>
</tr>
<tr>
<td>Ring</td>
<td>Point-to-point connections, simplified cabling, long path for back-to-back nodes, depending on protocol, the whole ring is used via shared medium (more complex access control)</td>
<td></td>
</tr>
<tr>
<td>Bus</td>
<td>Simplified cabling, direct communication (no routing), shared communication medium (more complex access control)</td>
<td></td>
</tr>
</tbody>
</table>

**Physical layer**

- **Propagation delay in a bus**
  - Time for a bit to traverse the full length of the bus (\( \delta \))

- **Limit to protocol efficiency in a bus**
  - Any message must be flushed from the bus before the next can be transmitted

- **Special cases**
  - Some protocols require spatial coherence at the bit level (\( \delta = 1 \))
    - \( \Rightarrow \) \( \text{Tx}_{\text{rate}} < \frac{1}{(2\delta)} \) (case of CAN)
Physical layer

- Special cases - wireless
  - In wireless networks the attenuation is strong (even worse with obstacles) and transmissions from one node may not reach all nodes in the network.

  ![RF signal strength diagram]

  Node 1 Node 2 Node 3 Node 4 Node 5

  1. Radius of the communication range

- Detection of collisions
  - In shared broadcast buses
    - If bit length > b, send 1 bit and listen
    - If bit length > b, it is not possible to relate the bit being transmitted with what is sensed on the bus. In this case a jamming signal is used.

  ![Collision diagram]

  Node 1 Node 2 Node 3 Node 4 Node 5

  1. Node 5 is already transmitting
  2. Node 1 does not listen to node 5 and starts transmitting
  3. Both transmissions interfere destructively at node 3

- Energy consumption
  - Influenced by: Transmission power, data rate, frame rate, network traffic, error control coding, retransmissions policy, type of medium access control and physical bit encoding.
  - Very complex relationship among the above parameters. Need to trade-off.
  - Highly relevant in autonomous systems, particularly in those expected to operate continuously for long term (wireless sensor networks).
  - Still a nice challenge!
Physical layer

- Energy consumption in wired networks
  - Some protocols use transceivers that are particularly energy expensive (e.g., CAN).
  - Reducing transmissions might be desirable in autonomous mobile robots
  - In some PIC18F458 based boards, overall current consumption raised from 70mA to 110mA corresponding to no CAN transmissions and constant transmission, respectively.
  - ...

Data link layer

- Addressing
  - Identification of the parts involved in a network transaction
    - Direct addressing
      - The sender and receiver(s) are explicitly identified in every transaction, using physical addresses (as in Ethernet)
    - Indirect (source) addressing
      - The message content is explicitly identified (e.g., temperature of sensor X). Receivers that need the message, retrieve it from the network (as in CAN and WorldFIP)
    - Indirect (time-based) addressing
      - The message is identified by the time instant at which it is transmitted (as in TTP)

- Logical Link Control
  - Deals with the transfer of network packets
  - Might have an impact on the network delay depending on whether sender/receiver synchronization is used
    - e.g., acknowledge mechanisms, retry mechanisms, request data on reply
    - Note that such synchronization using retries (PAR, ARQ) might have a strong negative impact on the data timeliness...
    - No point on continue trying to transmit out-dated values

Medium Access Control

- Master-slave
  - Access granted by the Master
  - Nodes synchronized with the master
  - Requires one control message per data message
Networks for Embedded Systems
©

Networks

M
e

✓ Master-slave
✓ The traffic scheduling problem becomes local to the master → good flexibility wrt scheduling algorithms
✓ Requires special mechanisms to handle aperiodic communication
✓ For high reliability the master must be replicated
✓ Master messages are natural synchronization points → supports precise tx triggering
✓ Ex: WorldFIP, Ethernet Powerlink, Bluetooth within each piconet

Medium Access Control
Controlled access

✓ Token-passing
✓ Access granted by the possession of the token
✓ Order of access enforced by token circulation
✓ Asynchronous bus access (generally impossible to determine a priori the exact access instants)
✓ Real-time operation requires bounded token holding time

Node 1

Node 2

Node 3

Node 4

Node 5

Time slot

Medium Access Control
Controlled access

Token-passing

TTTRT – Target Token Rotation Time fundamental configuration parameter that determines the time the token should take in one round, under heavy traffic

RTRT: Real Token Rotation Time time effectively taken by the token in the last round

ST – Synchronous time

THT – Token holding time

High tx jitter

Requires mechanisms to handle token losses

Similar to Round-Robin Scheduling

Ex: IEEE 802.4, FDDI, PROFIBUS

Medium Access Control
Controlled access

Token-passing

TTTRT – Target Token Rotation Time fundamental configuration parameter that determines the time the token should take in one round, under heavy traffic

RTRT: Real Token Rotation Time time effectively taken by the token in the last round

ST – Synchronous time

THT – Token holding time

High tx jitter

Requires mechanisms to handle token losses

Similar to Round-Robin Scheduling

Ex: IEEE 802.4, FDDI, PROFIBUS

Medium Access Control
Controlled access

✓ TDMA
✓ Time-Division Multiple Access
✓ Addressing can be based on time → High data efficiency
✓ Quality of synchronization bounds efficiency (length of guarding windows between slots)
✓ Typically uses static table-based scheduling
✓ Ex: TTP/C, FlexRay-static, TT-CAN, PROFINET

Medium Access Control
Controlled access

✓ TDMA
✓ Time-Division Multiple Access
✓ Addressing can be based on time → High data efficiency
✓ Quality of synchronization bounds efficiency (length of guarding windows between slots)
✓ Typically uses static table-based scheduling
✓ Ex: TTP/C, FlexRay-static, TT-CAN, PROFINET

Medium Access Control
Controlled access

TDMA

Time-Division Multiple Access

✓ Access granted in dedicated time-slot
✓ Time-slots are pre-defined in a cyclic framework
✓ Tight synchronization with bus time (bus access instants are predetermined)
✓ Requires global (clock) synchronization

Medium Access Control
Controlled access

CSMA

Carrier-Sense Multiple Access

✓ Set of protocols based on sensing bus inactivity before transmitting (asynchronous bus access)
✓ There may be collisions
✓ Upon collision, nodes back off and retry later, according to some specific rule (this rule determines, to a large extent, the real-time features of the protocol)
Medium Access Control
Uncontrolled access

- **CSMA/CD**
  - Carrier-Sense Multiple Access with Collision Detection
  - Used in shared Ethernet (hub-based)
  - Collisions are destructive and are detected within collision windows (slots)
  - Upon collision, the retry interval is random and the randomization window is doubled for each retry until 1024 slots (BEB - Binary Exponential Back-off)
  - Non-deterministic (particularly with chained collisions)

- **CSMA/CA**
  - Carrier-Sense Multiple Access with Collision Avoidance (async)
  - Access based on sensing bus inactivity during a synchronization interval (S-sync) plus a uniformly distributed random access interval with probability p (p-persistent)
  - Not collision-free (Ex: IEEE 802.11)

- **CSMA/BA (CA?)**
  - Carrier-Sense Multiple Access with Bit-wise Arbitration
  - Bit-wise arbitration with non-destructive collisions → Dominance protocol
  - Upon collision, highest priority node is unaffected. Nodes with lower priorities retry right after.
  - Deterministic (e.g. CAN)

- **CSMA/DCR**
  - Carrier-Sense Multiple Access with Deterministic Collision Resolution
  - Collisions are destructive
  - back off and retry based on priority (binary tree search)
  - Deterministic
  - Ex. G. Le Lann’s modified Ethernet

- **Switched**
  - micro-segmented network with central switching hub
  - Nodes send asynchronously messages to the switch using point-to-point links → no collisions
  - Contention at the network access is replaced by contention at the output ports (e.g. Switched Ethernet, ATM)
Network layer

✓ Logical Addressing

✓ Independence of higher protocols wrt the physical hardware
✓ Requires an ARP (address resolution protocol)
✓ Supports logical multicasts / broadcasts

✓ Ad-hoc routing

✓ Flat routing:
✓ All nodes have equal responsibility for maintaining routes and relaying messages

✓ Hierarchical / cluster-based routing:
✓ The network is broken into geographical areas, clusters, and the inter-cluster traffic is aggregated and routed as being a single message flow.

✓ Typical routing protocols include variations of the Dijkstra’s shortest path algorithm (based on flooding) as well as the distance vector technique.

✓ Routing

✓ Proactive:
✓ Routes are maintained continuously for all reachable nodes (uses periodic route dissemination plus)

✓ Reactive:
✓ Routes are established on demand for the duration of communication sessions, only. Optimization criteria can be used to guide the route set up, e.g. minimum energy, maximum power available, maximum link strength

✓ Geographic:
✓ Uses geographical information for routing purposes. Tries to establish routes that are as close as possible to the straight line between source and sink. Leads to shorter paths and consequently to lower end-to-end delays.
Network layer

- **Basic information propagation techniques**
  - **Flooding**
    - All nodes broadcast the messages they receive until a maximum number of hops or the destination is reached.
    - Simple and old technique. Presents:
      - low requirements for routing information (+)
      - too high traffic levels with many duplicates (-)
      - not resource-aware
  - **Gossiping**
    - All nodes send the messages they receive to one of its neighbors randomly.
    - Lower traffic levels but longer time to propagate data.

Network layer

- **Aggregation**
  - Nodes in a cluster send their inter-cluster traffic to the cluster head, that aggregates and relays it to the following cluster.
  - Reduces nodes involved in routing, thus reduces also routing information.
  - Data aggregation (combining data together) is another fundamental issue to reduce the amount of traffic in large sensor networks.

Protocol stack internals

- **Real-time protocol stacks may be based in:**
  - **Specialized communication controllers and/or custom device-drivers/stacks**
    - Low latency, high predictability, fine-grain control in queues, but ...
    - **Expensive**— non-COTS hardware, manpower for specific device-drivers development, longer development time, harder to debug, ...
    - **COTS hardware and software**
      - Cheap hardware, device drivers readily available for all the HW, full IP stacks supported, but ...
      - Potentially high latency, unpredictability, high resource consumption (memory and CPU), ...

Protocol stack internals

- **Using a TCP/IP stack:**
  - Easy connection to the intranet/Internet, use of standard tools/apps, easy development (app. code independent from HW), but ...
  - "standard" TCP/IP stack:
    - High CPU and memory consumption (code and data memory in the order of hundreds of KB, multiple data copies);
    - Not suitable to resource constrained embedded systems.
  - "lightweight" TCP/IP stacks (e.g. lwIP, uIP):
    - Code size is around 10KB and RAM size can be around 100's of B (suitable for 8/16 bit micro-controllers), efficient buffer management (zero copy), ...
    - Some limitations (e.g. single interface, single connection).

Protocol stack internals

- **Issues with general-purpose Device-drivers**
  - DD development is hard and costly
    - Many real-time systems use general-purpose DD (GPDD), eventually with some adaptations.
  - Issues with GPDD
    - Optimized for throughput (high latency/poor determinism)
    - FIFO queues between the host memory and the NIC internal memory
    - DMA/IRO optimizations conflict with predictability
      - Several messages can be buffered and generate a single INT/DMA transfer; Some operations have no defined time-bound
      - E.g. RTnet (Network stack for Xenomai and RTAI DD for 3COM NICS (rt_3c95x.c) is stated as "non real-time safe"

Protocol stack internals
Application layer

- Issues related with the application layer:
  - (middleware issues)
    - Cooperation model
      - Client-Server
      - Producer-Consumer
      - Producer-Distributor-Consumer
      - Publisher-Subscriber
    - Examples
      - CANopen
      - NDDS
      - CORBA

Application layer

- Cooperation model - Client-Server
  - Transactions are triggered by the receiver of the requested information (client).
  - Nodes that generate information are servers and only react to client requests.
  - The model is based on unicast transmission (one sender and one receiver).
  - Transactions can be synchronous (client blocks until server answers) or asynchronous (client follows execution after issuing the request).
  - e.g. CORBA, RMI, access to SDOs in CANopen

Application layer

- Cooperation model – Producer-Consumer
  - Transactions are triggered by the nodes that generate information (producers).
  - The nodes that need the information, identify it when transmitted and retrieve it from the network (consumers)
  - The model is based on broadcast transmission (each message is received by all nodes)
  - e.g., PDOs distribution in CANopen, many CAN higher layers

Application layer

- Cooperation model – Producer-Distributor-Consumer
  - Basically similar to Producer-Consumer
  - Transactions are triggered by a particular node, the distributor, upon request from the producers or according to a pre-established schedule.
  - It is an implementation of PC over master-slave
  - e.g., WorldFIP, sync traffic in FTT-CAN

Application layer

- Cooperation model – Publisher-Subscriber
  - Elaborate version of Producer-Consumer using the concept of group communication
  - Nodes must adhere to groups either as publisher (produces information) or as subscriber (consumes information)
  - Transactions are triggered by the publisher of a group and disseminated among the respective subscribers, only (multicast).
  - e.g., DDS, sync traffic in FTT-SE

Application layer

- Some Application Layer examples
  - CANopen
  - NDDS
  - CORBA
CANopen

- CANopen
  - communication and application standard for distributed systems
  - Maintained by the CAN-in-Automation (CIA) group
  - http://www.can-cia.com
- Key features:
  - Transmission process data according to the producer/consumer model
  - Standardized device description (data, parameters, functions, programs)
  - Standardized access to device parameters
  - Standardized services for device monitoring (e.g., membership functions based on heartbeat)
  - System services: synchronization message, central time-stamp message (e.g., synchronous data acquisition)
  - Emergency messages
- Adopted by other protocols (e.g., Ethernet POWERLINK)

CANopen protocols

- Process Data Objects (PDO)
  - Carry actual application data; broadcast, producer/consumer cooperation model, unacknowledged
- PDO types
  - Asynchronous PDOS: event-controlled, automatically transmitted when at least one of the process variables mapped in a PDO is altered
  - Synchronous PDOS: transmitted after reception of synchronization message (Sync Object).
    - Transmission is carried out synchronously in the entire network.
    - All device inputs are sampled on the arrival of the sync object
      - Sampled data transmitted after the next sync message (1 cycle constant delay)

CANopen

- Device configuration (service data objects/ SDO)
  - SDOs are used to read/write device OD entries
  - Client/server, logical 1:1 channel
  - Acknowledged service: each client SDO requires a server answer
- Network management (NMT)
  - Network management services for control of the communication state of network nodes and node monitoring
    - Node states: Initialization / Pre-operational (used for configuration, no PDO transactions) / Operational / Stopped (only heartbeat/guard messages can be transmitted)
    - Network master may change the node’s state

DDS

- DDS
  - Data Delivery Service (DDS)
    - Middleware for distributed real-time applications made by Real-Time Innovation, Inc.
    - Based on the Real-Time Publisher-Subscriber model
    - Standardization within the Object Management Group (OMG) DDS database shared among all nodes, which have an holistic view on the communication requirements
    - Publishers create “topics” (e.g. temperature) and publish “samples”
    - The network delivers the “samples” to each subscriber
    - Addressing, delivery, flow control, etc. handled by the DDS

CORBA

- Common Object Request Broker Architecture
  - Open specification proposed by the OMG
  - http://www.omg.org
  - Objective: Clients use remote objects as if they were local
  - Characteristics:
    - Multiplatform: Windows, Linux, Unix, MacOS, ...
    - Multilingual: Ada, C, C++, Java, Python, ...
    - Interoperability between languages and platforms
    - Multiple vendors (some are freeware products)
Part 4
Some related network protocols

WorldFIP

- MPS – Messagerie Periodique e Sporadique
- Producer-Distributor-Consumer model
- Concept of Network Variable
  - Entity that is distributed (several local copies exist in different nodes)
  - Can be periodic or aperiodic
  - Local copies of periodic variables are automatically refreshed by the network
  - Local copies of aperiodic variables are refreshed by the network upon explicit request

WorldFIP

- Elementary cycles organized in phases:
  - Periodic (P1)
  - Aperiodic (P3)
  - MMS messages (P2)
  - Sync - padding (P4)

Each phase is a server that can schedule a class of traffic

TTP/C

Time-Triggered Protocol (TTP/C – for SAE class C)

- Created around 1990 within the MARS project in the Technical University of Vienna
- Aims at safety-critical applications
- Considers an architecture with nodes integrated in fault-tolerant units (FTUs), interconnected by a replicated bus
- Includes support for prompt error detection and consistency checks as well as membership and clock synchronization services.
**TTP/C**

- Network and nodes architecture

![TTP/C Diagram]

**PROFIBUS**

PROCeSS FieldBUS

- Created in the late 80’s by Siemens, in Germany
- Two main application profiles:
  - PROFIBUS / FMS - Fieldbus Message Specification
  - PROFIBUS / DP - Decentralised Peripherals
  - The most common profile (~90%)
- Data payload between 0 and 246 bytes
- Direct-addressing (1 byte, possibly extended)
- Hybrid bus access control
  - Token-passing among masters
  - Master-Slave in each individual data transaction

![PROFIBUS Diagram]

**PROFIBUS**

- Traffic pattern under maximum load:
  - Equal distribution of bandwidth among N nodes
  - Intervals of bus inaccessibility = TTR*(N-1)

![Traffic Pattern Diagram]

**PROFIBUS**

- Recent support for: (PROFIBUS / DP-v2)
  - Isochronous traffic
    - Equidistant token arrivals forming cycles
    - Cycle synchronization (specific global frame - broadcast)
    - Two windows in the cycle (isochronous and async traffic)
  - Direct slave to slave communication
    - Producer – Consumer style
    - Specifically developed for high accuracy drives

![Recent Support Diagram]
CAN
Controller Area Network

✓ Created in the early 90s by Bosch, GmbH, for use in the automotive industry.
✓ Data payload between 0 and 8 bytes
✓ Source-addressing (message identifiers) with 11 bits in version A and 29 bits in version B
✓ Asynchronous bus access
✓ Non-destructive arbitration based on message identifiers (establish priority)
  ✓ Bit-wise deterministic collision resolution
  ✓ CSMA/BA (CA?, DCR?)

Ethernet

✓ Created in the mid 70s (1) by Robert Metcalfe at the Xerox Palo Alto Research Center.
✓ CSMA/CD non-deterministic arbitration (used in shared Ethernet, only)
✓ 1-persistent transmission (transmits with 100% probability as soon as the medium is considered free)
✓ Collisions can occur during the interval of one slot after start of transmission (512 bits)
✓ When a collision is detected a jamming signal is sent (32 bits long)
✓ Frames vary between 64 (min) and 1518 (max) octets

Motivations for using Ethernet

✓ In application domains beyond its original one
  e.g. factory automation, large embedded systems
  (Deutsch, 2001)
✓ Cheap wrt other high speed technologies
✓ Widely available
✓ Scalable tx rate
✓ High bandwidth
✓ Easy integration with office networks (important for logging, management and multi-level integration, e.g. CIM)
✓ IP stacks widely available
✓ Well known and mature technology

Ethernet

✓ Thrashing effect
  As the offered load increases, the actual load handled by the network (throughput) decreases

Maximum network-induced delays can be very long with high packet-drop rates
Networks

Ethernet

- **Downside of using Ethernet** in application domains beyond its original one, e.g., factory automation, large embedded systems (Degroote, 2001)
  - Connection costs higher than traditional fieldbuses (PHY, transformer, MAC)
  - High communication overhead (for short data items)
  - High computing power requirements (for efficient bandwidth usage)
  - Typical star topology not always adequate (may lead to extra long cabling)
  - Existing protocol stacks (mainly IP) not optimized for real-time operation

- Existing protocols not optimized for efficient bandwidth usage (PHY, transformer, MAC)

- **Existing protocol stacks** (mainly IP) not optimized (for efficient bandwidth usage)

- **Control transmission** (e.g., token-passing (RETHET, RT-EP), master-slave (FFT-Ethernet, ETHERNET-Powerlink), TDMA (PROPNET-PIR, ...))
  - Micro-segment the network using switches (IEEE 802.1D) – Avoids collisions!

- **Making Ethernet real-time**
  - Keep network load low (~1%, e.g., NDDS)!
  - Avoid bursts (e.g., traffic smoothing)
  - Modify the back-off and retry mechanism (e.g., CSMA/DCR, Virtual-time, windows, EDCF)

- **Current mode**;
  - Reduced collisions

- **New mode**;
  - Reduced collisions

- **Beacon mode**

- **Superframe with GTS** (Guaranteed Time Slot)

- **Superframe without GTS** (Guaranteed Time Slot)

- **ZigBee**

- **WiFi (802.11)**
  - Created in the 1990s for the SOHO domain. As Ethernet, 802.11 became widely used as wireless communication infrastructure.
  - 3 modes/bands: 802.11b/g (ISM-2.4GHz), 802.11a (5GHz)
  - Scalable Tx rates between 1Mbit/s and 54Mbit/s
  - Many mechanisms to reduce collisions and hidden-terminals
  - Growing interest for QoS support (initial PCF mode was abandoned, currently there is a new mode, 802.11e)

- **FlexRay**
  - Created in the early 2000s by an industrial consortium from the automotive domain.
  - First public specifications available since 2004 (several changes before that)
  - Is expected to replace CAN and contend with TTP/C
  - Tx rate up to 10Mbit/s

- ZigBee

- **ZigBee**

- **WiFi (802.11)**
  - Created in the 1990s for the SOHO domain. As Ethernet, 802.11 became widely used as wireless communication infrastructure.
  - 3 modes/bands: 802.11b/g (ISM-2.4GHz), 802.11a (5GHz)
  - Scalable Tx rates between 1Mbit/s and 54Mbit/s
  - Many mechanisms to reduce collisions and hidden-terminals
  - Growing interest for QoS support (initial PCF mode was abandoned, currently there is a new mode, 802.11e)

- **FlexRay**
  - Created in the early 2000s by an industrial consortium from the automotive domain.
  - First public specifications available since 2004 (several changes before that)
  - Is expected to replace CAN and contend with TTP/C
  - Tx rate up to 10Mbit/s

- **Superframe with GTS** (Guaranteed Time Slot)
  - **Superframe without GTS** (Guaranteed Time Slot)
Part 5
Some related on-going research

FTT-SE
- Keeping under control the traffic load submitted to a switched network
- Schedule traffic per cycles
- Submit only the traffic that fit in a cycle
- Eliminate memory overloads
- Support full priority scheduling
- Support server-based scheduling

QoS adaptation with FTT-SE

Server-SE

Resource-reservation switch
- Providing timeliness, flexibility and high robustness in switched Ethernet networks
- Enforce negotiated channel characteristics (policing)
- Reject abusive negotiated traffic (filtering)
- Confin e non-negotiated traffic to separate windows (selection)
Resource-reservation switch

- 1 real-time aperiodic stream (switched on and off)
- 1 non-real-time stream (bursty and persistent)
- Large file transfer
- Sent to the same output link

Robust IEEE802.11 communication

- Providing robust IEEE 802.11 communication for teams of autonomous mobile robots
  - Reconfigurable and adaptive TDMA to cope with uncontrolled traffic

FTT-CAN

- Providing timeliness (synchronization) and flexibility in embedded CAN networks
  - Motion
  - Odometry

Relative localization with IEEE802.15.4

- Using IEEE 802.15.4 communication to
  - Discover and track the topology of a team of autonomous mobile robots
  - Find their relative positions for team coordination

Conclusion

- The network is a fundamental component within a distributed system (supports system integration)
- Real-time operation requires time-bounded communication
- Appropriate protocols must be used
- Growing distribution, integration and flexibility in distributed embedded systems require more support from the network to maintain temporal guarantees, safety and robustness
- Still many open issues in trying to improve the timeliness of the communication, mainly in multi-hop or multi-segment scenarios together with reconfigurability and adaptivity, and high dependability (reliability, availability, security, …)

Bibliography

Other suggested reading


**Announcement**

- CyberRescue@RTSS2009
- [http://robot.unipv.it/cyberrescue-RTSS09](http://robot.unipv.it/cyberrescue-RTSS09)
- Control the team of 5 robots with ad-hoc communication capabilities to reach the victim in the least time